Sam Buchwalter
David Mendelssohn
CMSC 122

# Playlist Request Implementation: buchwalters16-dmendelssohn

## Goal:

We decided that the focus of our project was to create something that was both interesting and practical. Thus, we want to create a program which, given a party playlist and a few inputs about the party, will determine if a song request fits in the playlist and adds it at an appropriate location in the playlist queue. Inputs would include arguments such as request info and departure time of the requester. The function will determine, using data from the playlist on event duration and event type (i.e. “Only 90s Kids Will Remember These Songs,” “RIP in Peace David Bowie,” or, perhaps, a sort of “energy level” of the playlist) if and where a song can fit in the playlist before the requester would depart. The program would then return whether the song has been added to the playlist and in how many minutes one should expect the song to play, as well as surrounding songs. Additional functionality can be added if necessary.

## Data:

We will be using the Million Song Dataset (MSD), which can be located at http://labrosa.ee.columbia.edu/millionsong/. This song is a compilation of tags, user data, lyrics, genres, years, and more. Most importantly, the dataset includes fields such as loudness, danceability, popularity of the song/artist, song energy, similar artists, and song duration.

## Tasks:

We need to create a function that will pull data from the MSD and analyze the response given. Then, we will need a function that will take the playlist and the inputs from the playlist creator and determine what attributes are most important for the playlist. Once we have those, we will need a process that will screen songs from the requester based on MSD data (this will be the hard part). To do this, we will have to figure out how to use the MSD/EchoNest API in an efficient way. After we screen the songs, we will need to place them in the playlist based on the other songs on the playlist. We will then return the requester information of whether the song was accepted and if accepted, when it will play.

**Timeline**

Fifth Week: Determine how to pull information from the MSD; create a useful data structure from user input

Sixth Week: Create the function which pulls song information

Eighth Week: Create function which determines if the song fits in the playlist

Ninth Week: Create function to determine where to place song and return result

Tenth Week: Optimize

**New Items:**

MSD Reader: code to access/read MSD or EchoNest API

SWIG (option 1): To accelerate and optimize MSD traversal and song analysis process

Fuzzy Matcher (option 2): fuzzywuzzy/Levenshtein fuzzy matching for misspelled inputs

7digital API (optional 3): fetch a portion of the playlist to confirm song choices via samples